Assessing the Security of an Industrial Hatchery using the FAST-CPS Framework

Laurens Lemaire
Jan Vossaert
Bart De Decker
Vincent Naessens

Report CW 710, December 2017

KU Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

Assessing the Security of an Industrial Hatchery using the FAST-CPS Framework

Laurens Lemaire
Jan Vossaert
Bart De Decker
Vincent Naessens
Report CW 710, December 2017

Department of Computer Science, KU Leuven

Abstract

Cyber-physical systems (CPS) are used to control and monitor a nation's critical infrastructure. Unfortunately the security of these systems is often lacking. They were not designed with security in mind and often contain legacy components with known vulnerabilities. System owners are often not aware that these vulnerabilities are present, or what their impact may be on system operations when they are exploited. To help the system owners with these issues, the FAST-CPS framework has been developed.

FAST-CPS offers an automatic security analysis of cyber-physical systems based on a model of the system: Vulnerabilities are extracted and their effects on the system processes are returned, a data flow analysis is performed to identify conflicts between the different stakeholders in the system, and attack trees are automatically constructed for selected attacker goals in order to help with a qualitative risk assessment of the system. This article demonstrates the workings of the FAST-CPS framework on a real-life case study: an industrial hatchery.

Assessing the Security of an Industrial Hatchery using the FAST-CPS Framework

Laurens Lemaire[1], Jan Vossaert[1], Bart De Decker[2] and Vincent Naessens[1]

[1] KU Leuven, MSEC, iMinds-DistriNet, Department of Computer Science
Gebroeders Desmetstraat 1, 9000 Ghent, Belgium
firstname.lastname@cs.kuleuven.be
[2] KU Leuven, iMinds-DistriNet, Department of Computer Science
Celestijnenlaan 200A, 3001 Heverlee, Belgium
firstname.lastname@cs.kuleuven.be

Abstract. Cyber-physical systems (CPS) are used to control and monitor a nation's critical infrastructure. Unfortunately the security of these systems is often lacking. They were not designed with security in mind and often contain legacy components with known vulnerabilities. System owners are often not aware that these vulnerabilities are present, or what their impact may be on system operations when they are exploited. To help the system owners with these issues, the FAST-CPS framework has been developed [19].
FAST-CPS offers an automatic security analysis of cyber-physical systems based on a model of the system: Vulnerabilities are extracted and their effects on the system processes are returned, a data flow analysis is performed to identify conflicts between the different stakeholders in the system, and attack trees are automatically constructed for selected attacker goals in order to help with a qualitative risk assessment of the system. This article demonstrates the workings of the FAST-CPS framework on a real-life case study: an industrial hatchery.

Keywords: Cyber-Physical Systems, Security Assessment, FAST-CPS, Attack Trees

1 Introduction

Cyber-physical systems (CPS) comprise a broad range of applications, including industrial control systems (ICS), robotics, supervisory control and data acquisition (SCADA) systems, and more. A CPS is a network of interacting elements with physical input and output, it usually contains various remote field sites where a certain process is taking place. Each field site consists of sensors and actuators, controlled locally by a programmable logic controller (PLC), remote terminal unit (RTU), or similar device. These remote sites are connected to a centralized control network where operators can remotely monitor and control the processes. The combination of computational and physical aspects is what defines CPS [14,35].

In the past decades, these systems have evolved from proprietary, isolated systems to complex interconnected systems that are remotely accessible and often use commercial off-the-shelf (COTS) components. This has made them easier to use, but also easier to attack [6]. These systems have different security requirements from traditional IT systems [4,29]. In IT, the CIA triad is often referenced: Confidentiality, Integrity, Availability [10]. IT systems are often used to deal with personal data or other privacy sensitive information, hence the confidentiality of this data is paramount. Encryption and cryptography have been studied a great deal to ensure this requirement [9]. For similar reasons, it should not be possible to tamper with the data that is sent through these systems. The integrity of the data must be preserved, and it should be possible to prove that the data has not been tampered with. This is achieved with hash functions or MACs [2]. Finally, the availability of the system is a third requirement. In case of emergencies, back-up centres should be in place to ensure continuous operation.

For CPS the situation is different [26,23]. These systems often take care of critical infrastructure, e.g. traffic lights, power plants, water treatment facilities, etc. Hence availability is suddenly much more important. The consequences of one hour downtime for a traffic light system can be catastrophic. Confidentiality, on the other hand, is often less important than in IT systems. The data that flows through these systems is usually just sensor data and actuator commands. As a result, some communication protocols commonly used in CPS do not provide encryption. Hence some papers reverse the CIA triad and talk about AIC in control systems [5]. There are exceptions to this, for instance production environments where some data is to be kept confidential. Here encryption is seen as a possible solution, though only if it does not adversely affect the availability constraints.

The physical nature of the systems also introduces safety as a new factor that must be considered. When CPS malfunction, people can get hurt, in IT systems this is rarely the case. Finally, these systems should be highly deterministic and reliable, even the smallest chance of an error or unexpected behaviour should not be allowed. Hence in CPS the SRA triad is also used: Safety, Reliability, Availability [34].

Numerous cyber-physical systems have been attacked in recent years. Examples of notable incidents include the Maroochy Shire sewage spill in Australia [1], and the Stuxnet worm in Iran [22,13,7]. More recently, the Ukrainian grid was compromised which caused hundreds of thousands of people to lose power for several hours [36,3], and the WannaCry ransomware attack affected control systems worldwide [25]. In the latter two cases, known vulnerabilities were exploited by the attackers, highlighting the importance of identifying known vulnerabilities in these systems and evaluating the possible consequences of an attacker exploiting them.

Various research initiatives have been undertaken in previous years to improve the security of cyber-physical systems, both from the academic world and industry. One such initiative is the FAST-CPS framework [19]. This framework aims to provide an automatic security assessment of CPS based on a CPS model.

In this article the functionality of the FAST-CPS framework is briefly explained, and the framework is then applied to a real-life case study. Step by step, it is shown which actions the assessor must perform and what kinds of feedback the platform can provide.

The outline of this article is as follows. Section 2 discusses related work in the area, including other tools that perform a security assessment of CPS. Section 3 gives an overview of the functionality of the framework. In section 4, the framework is then used to analyse the security of an industrial hatchery. Section 5 evaluates the framework. Finally, section 6 concludes the article.

2 Related Work

In [28], the authors use UML system models to automatically create dynamic fault trees (DFTs). These DFTs are then analysed to compute the reliability of a system. FAST-CPS applies a similar approach. The Systems Modelling Language (SysML) is used instead of UML. SysML extends UML with several new or modified diagrams and is more suited for modelling systems or systems-of-systems. Further, the approach does not aim to draw conclusions regarding reliability, but focuses on security instead. Currently there is a lack of attention for system security in model-based system engineering, as discussed in [27]. FAST-CPS aims to fill this gap.

Other tools for the automated security assessment of cyber-physical systems are under development, aiming to assist a system operator with identifying whether their system is secure. Four such tools that are complete or nearing the stage of completion and that are publicly available are presented here:

2.1 CSET

Homeland Security has created the Cyber Security Evaluation Tool (CSET) [11]. This tool checks compliance of a system with a chosen standard through a question and answer method. There are 24 standards the assessor can choose between. Once a standard has been chosen, the tool generates a list of questions that will assess system compliance with the given standard.

CSET also has a diagram feature where the assessor can model the network topology of their system. The tool then gives warnings in order to assist the assessor in finding a more secure network architecture. From this diagram, the assessor can also generate a list of questions from a standard, but only questions related to the components in the system. Hence this feature helps in condensing the list of possible questions.

2.2 ADVISE

ADVISE [20] determines which way an attacker is most likely to go about attacking a system. The tool has been added to the Möbius framework [8] to make use of its modelling formalisms and solution techniques.

Modelling a CPS in ADVISE is done from the point of view of the attacker. The system architecture itself is not modelled, only the attacker path. First the assessor must decide what the attacker goals are. Next, the assessor adds the *attack steps* that the attacker must complete before the goals are reached. Each attack step is then assigned further prerequisites. There are three possible types: *access* to networks or workstations, required attacker *skills*, and *knowledge* such as passwords. The result of an attack step can be an attacker goal, or increased access/knowledge for the attacker.

When the assessor has mapped out all the attack steps, he now has to model the adversary. This includes specifying the attacker's initial knowledge/access and the attacker proficiency in all skills. The assessor also indicates how much the attacker cares about detection versus pay-off, and the relative importance of the different goals. The result is an Attack Execution Graph (AEG). The AEG represents potential attack steps against the system. ADVISE automatically generates an executable model that represents how the adversary is likely to attack the system. Once the attack execution graph is modelled, the assessor can run the framework to determine which attacker goals will be reached with what probability.

2.3 CyberSAGE

CyberSAGE [33] is similar to ADVISE in terms of output. The assessor will have to specify a workflow of the threat agent, and the result will show probabilities of the attacker reaching his goal and his attack steps. However, the tool requires less security expertise to use as the attack steps are fixed and there are less attacker variables to decide on.

To generate the resulting argument graph, the assessor needs to provide four pieces of input. First there is the workflow graph which specifies the actions of the attacker. There is a list of 19 possible actions the assessor can choose from, including *send command*, *physical access*, *inject malicious messages*, and so on. The assessor has the possibility to add additional actions in the rules engine.

Next, the assessor models the system components and networks. This is largely similar to the diagram feature of CSET, with the exception that components can have various properties regarding authentication, access control, encryption, etc. Less components are available, but the assessor can add his own components and properties to the palette.

Then, the assessor must model the attacker. The attacker attributes include his skills, his intention, his access to the system, and his resources. The attacker model is less detailed than the one used in ADVISE.

The final element of input is the rules engine. A default rules file is supplied with each CyberSAGE instance. The assessor has the option to edit this and tweak the probabilities, or add additional attacker actions.

2.4 CySeMoL

CySeMoL [30,31] provides the same kind of feedback as the two previous tools, e.g. the probability of an attacker reaching some attack goals in a system. However, in CySeMoL both the attacker and the attacker goals are fixed, the assessor cannot change them. For their attack probabilities, CySeMoL assumes that the attacker is a penetration tester who only has access to public tools.

Using CySeMoL does not require security expertise from the assessor. He just has to model his system according to the Probabilistic Relational Model (PRM) employed by CySeMoL. The PRM specifies a theory on how attributes in the model depend on each other.

When modelling the network architecture, the available components on the palette do not include typical CPS components, but rather a general *OperatingSystem* block that can be used for all PLCs, HMIs, and so on. These can then be connected with the relevant *ApplicationServer* or *ApplicationClient* components. Each component has a fixed number of attributes linked to it, for instance the *ApplicationClient* component has an attribute *HasAllPatches* which the assessor can change to false or true. Modelling a CPS in CySeMoL is not as straightforward as with the other tools, but there are several tutorials available.

These four tools all aim to help a system operator decide whether their system is secure, but they go about it in different ways. None of the tools attempt to find known vulnerabilities in the system, which is the one of the main purposes of FAST-CPS. Research by Gartner shows that up to 75% of attacks on these systems is caused by exploiting known vulnerabilities for which a patch or secure configuration is already available[21].

As a result, the different tools can complement each other. For example, FAST-CPS can be used to find known vulnerabilities in the system and their effects on system security, this new information can then lead to a more detailed attack path in the ADVISE tool in order to get more accurate feedback about attack probabilities.

What the tools have in common is that they only find weaknesses in the system, they do not suggest solutions. An exception is the CSET diagram feature that can give some very basic network topology feedback, i.e. place a firewall here. Hence the system operator may still need to contact a security audit company if he does not know how to fix a vulnerability in the system.

3 FAST-CPS

FAST-CPS is an extension of the Papyrus Eclipse modelling environment. Papyrus allows users to model complex systems using the SysML modelling language. FAST-CPS has extended this framework with a knowledge base system. A parser converts the SysML diagrams into input for the logic engine, where various types of feedback are automatically extracted. FAST-CPS differs in required input and returned feedback from the tools presented in the previous

section. ADVISE and CyberSAGE require significant security expertise as the assessor is expected to model the workflow of the attacker. In FAST-CPS, this workflow or attacker strategy will be one of the outputs of the system, and no security expertise is required. Furthermore, the feedback of the framework goes beyond the probability of a certain attack succeeding: The framework will identify known vulnerabilities at the hardware/software level and return their effects on the system processes. In addition, the FAST-CPS framework also provides some data privacy feedback, showing an assessor what happens with the data assets in his system and returning any conflicts this causes with the data preferences of stakeholders. A full comparison of FAST-CPS and these four other tools has been published at [17]. The functionality of the FAST-CPS framework can be split up in three main parts:

Extracting Process Vulnerabilities. The assessor will model his cyber-physical system in the Systems Modelling Language (SysML). The framework will then perform an automatic security analysis of the system. Vulnerability databases such as the one from ICS-CERT are incorporated in the framework in order to find known hardware and software vulnerabilities. The effect of these vulnerabilities on the system process is then computed. The assessor can also perform simulations to reason about hypothetical scenarios. This methodology is explained in detail at [19].

Performing a Data Flow Analysis. The assessor can also reason about the flow of data assets in his system. With multiple interacting stakeholders present in today's CPS, it is no longer trivial to know what is happening to your data. The FAST-CPS framework will return an asset profile which presents an overview of which stakeholders can perform operations on which data assets. If this results in conflicts with any stakeholder preferences, these will also be returned. An assessor can specify multiple actors for any given stakeholder role, and the system will search for configurations of actors that cause the least number of conflicts. More information about this methodology can be found at [18].

Towards Qualitative Risk Assessment. Finally, the framework will also automatically construct attack trees based on the provided system model and a chosen attacker goal. The assessor can model different attackers and evaluate the trees with respect to a chosen attacker. The system will then return the relative difficulty for the attacker to reach his goal, and show the optimal attacker strategy that achieves it. Based on this information, an assessor can identify the weak parts of his system and strengthen them accordingly. By assigning impact values to the attacker goals, a qualitative risk assessment can be performed. This work has been accepted for publication at the 2017 International Conference on Critical Information Infrastructures Security.

4 Assessment of a Cyber-Physical System using FAST-CPS

In this section, the full functionality of FAST-CPS will be showcased on a case study: an industrial hatchery. The aim is to give a detailed overview of all the

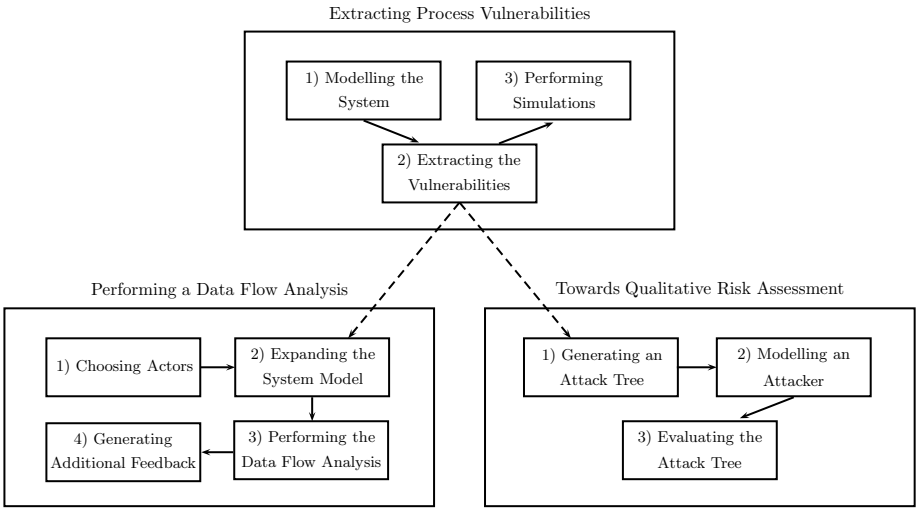

Fig. 1. An overview of the steps required to use FAST-CPS.

necessary steps so system owners can start using the framework. The steps required to use the framework are shown in Figure 1.

4.1 An Industrial Hatchery

The Cyber-physical system considered in this case study is displayed in Figure 2. It consists of a process network, a de-militarized zone (DMZ), and a company network. The process network is an industrial hatchery. The DMZ contains a data historian where process data from the hatchery is sent. Employees in the company network can then access the DMZ to retrieve this data. No direct communication between the company network and process network is possible. The company network is connected to the internet. Customers can contact the company to place orders. A cloud-based scheduler is used to process these orders and create a schedule that makes optimal use of the available incubators in the hatchery. The schedule is pushed to a PC in the DMZ and operators from the hatchery can access it there. Finally, the equipment in the hatchery is provided by an equipment vendor that offers a predictive maintenance (PdM) service. LoRa sensors measure various incubator metrics and send these over a telecommunications cloud to the PdM cloud used by the equipment vendor. The vendor is also able to send firmware updates for the PLCs and touch screens to the SCADA PC.

The process network is a hatchery that consists of two types of incubators: *setters* and *hatchers*. Each incubator consists of various sensors and actuators that are connected to a PLC. At the front of the incubator a touch screen is used for monitoring the environment variables and controlling the actuators.

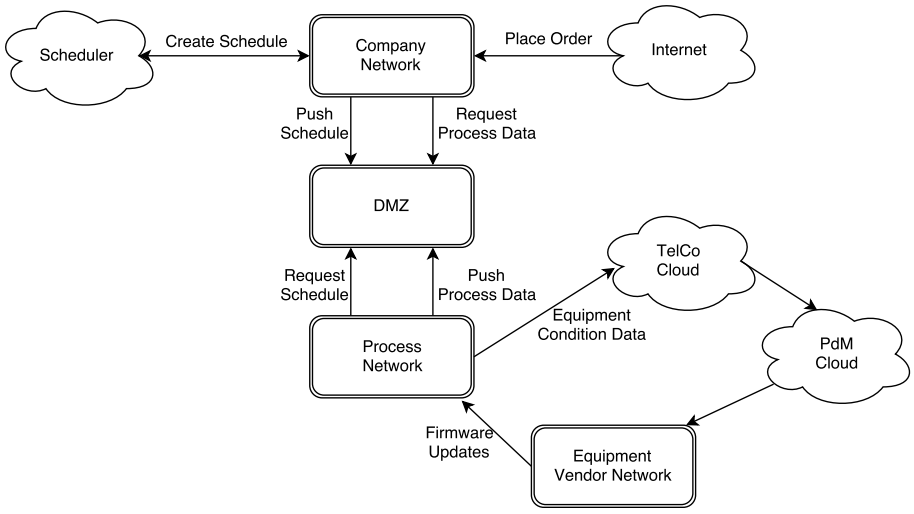

Fig. 2. An overview of the use case architecture.

The manager's office contains a central switch which is connected to all the PLCs. Here we also find an eWON router that is used by the equipment vendor to connect to the hatchery remotely, as well as a SCADA PC that logs all the data and can be used to control all incubators. The process network utilises role based access control. There are currently three different types of users in the hatchery. The least privileged users are the technicians, they are not able to change any process parameters. Next are the operators, they can change all parameters of the incubators, including the temperature settings, humidity, CO2 levels, etc. The local managers make up the final level. They have access to the SCADA PC and can apply updates to the machines. Other users considered in the case study are an Employee in the company network, a Vendor in the equipment vendor network, and an Attacker.

4.2 Extracting Process Vulnerabilities

A first feature of the FAST-CPS framework is the automatic extraction of process vulnerabilities. This functionality is limited to the process network and the DMZ as the vulnerability databases that are incorporated into FAST-CPS are specific to CPS components.

STEP 1: Modelling the system. The first step for the assessor is to model the process network and the DMZ of his system. This can be done in the framework using the SysML diagram language. Full SysML functionality is provided by switching to the Papyrus perspective. Detailed instructions on how to model a CPS correctly for security evaluation with the framework are given

at [15]. The model is created manually by the assessor, hence the availability constraints of the CPS are not threatened. Below some of the important diagrams will be highlighted, referenced figures can be found in Appendix A.

Internal Block Diagrams. The assessor will start by modelling all the hardware components and the connections between them. For each hardware component, the assessor models the product and version information. This is necessary to allow feedback from vulnerability databases to be taken into account. Next, the software modules in each component are modelled, an internal block diagram is used for this purpose. An example of a diagram representing a setter touch screen is given in Figure 5. The touch screens used are from the Schneider Pro-Face GP-4401 series, this information is added in the form of a comment. The touch screen contains two software modules: an authentication module and a control module that can change the state of 2 *process parameters*: a process actuator and an upper bound in the process invariant. The control module can only be reached after authenticating to the authentication module. Additional authentication and authorization information is modelled in separate diagrams.

Parametric Diagrams. Parametric diagrams are used to model the system processes. A system process is modelled as an invariant that expresses relations between environment variables, thresholds and the status of actuators. For instance, one of the processes in our industrial hatchery specifies that the temperature inside a setter incubator should not exceed an upper bound of $37.5°C$. If it does, the cooling valve actuator in the setter should be opened. This process is modelled in the parametric diagram shown in Figure 6. All relevant parameters are included in the centre process block. S_{1T} represents the temperature of setter 1, S_{1V} represents the status of the cooling valve and S_{1UB} represents the upper bound. The invariant is provided as the constraint of the process block. Dependency arrows between parameters indicate that changing a parameter might also change the state of another, i.e. in this example changes in both the temperature and the upper bound could affect the status of the valve. Similar diagrams are created for the other processes in the process network.

Block Definition Diagrams. A final highlighted modelling concept is the *policy specification*, for which a block definition diagram is used. The policy specification indicates which user groups are allowed to perform certain operations on the process parameters. Part of the security evaluation will be to determine whether this policy is violated or not. Figure 7 shows part of the diagram. Four SysML Actor components represent the user groups in the system: the technician, the operator, the manager, and an attacker. Two parameters have been modelled as blocks, and the possible operations have been added: Read() and Modify(). Now usage arrows are drawn between the users and the operations, indicating that a user is allowed to perform said operation on the parameter. The attacker and the technician are not allowed to perform any operations, the operators and managers can read/modify the status of the setter valve and read the setter

temperature.

STEP 2: Extracting the Vulnerabilities.

Once the system is modelled, the security analysis can be performed. To do this, the assessor must switch to the Java perspective in the framework and right-click the .di file of his system model. A FAST-CPS menu will be available from where the security analysis can be launched. When chosen, the .uml file of the system model will be parsed into input for our logic programming framework, IDP3, which will perform the evaluation. The details of how this evaluation is performed are not in the scope of this article, however they can be found at [19]. The result of the security evaluation is three new windows that open in the framework editor. The first window contains the input given to IDP, i.e. the result of parsing the SysML model into an IDP-readable file. The second is the output of the IDP analysis, containing the actual results. This file is hard to read as it is just a collection of tuples, hence the third window shows a formatted version of the output that focusses on the information extracted from the important predicates and functions.

The formatted file starts by listing any component or module vulnerabilities found in the system model. These are found by incorporating CPS vulnerability databases in our logic framework. An explanation of how these vulnerabilities are extracted from the model can be found at [19]. Next, the file lists any compromised system tasks. There are three types of tasks: Storage tasks concern storage of process parameters in components. Control tasks connect software modules with the actuators they can control. Communication tasks concern the forwarding of parameters to their destination. The logic theory infers which tasks are compromised as a result of the found vulnerabilities. Finally, at the bottom of the file any failed access policy verification queries are listed, and an example for why they fail is included. The different access policy verification queries are:

- Only authorized user groups are able to modify parameters
- Only authorized user groups are able to read parameters
- Only authorized user groups have access to protected networks
- Users only possess the credentials they are authorized to possess
- Only authorized user groups have physical access to components marked as critical

The full logic theory can be found at [15].

In the case study, two types of component and module vulnerabilities are reported: The SCADA PC operating system module contains a vulnerability that can lead to a Remote Code Execution and the setter PLCs contain data leakage and denial of service vulnerabilities. In the IDP Output file this is reported as follows:

- *HasVulnerability(SCADAPCOSModule,RemoteCodeExecution)*
- *HasVulnerability(Setter1PLC,DataLeakage)*
- *HasVulnerability(Setter1PLC,DoS)*

Upon inspection, the remote code execution is caused by the SMB protocol inside the Windows OS on the SCADA PC. This is the vulnerability that allowed the WannaCry ransomware to spread [24]. The setter PLCs are ControlLogix PLCs from Rockwell Automation which have not yet been patched to the latest firmware version, which leads to DoS and data leakage vulnerabilities [12]. Figure 8 shows the formatted output returned by the security analysis. The results of the component and module vulnerabilities are visible among the compromised tasks. The setter PLC does not perform its storage tasks correctly and it may not be possible to retrieve the temperature parameter from the sensor, which is a compromised communication task. Likewise, the SCADA PC control module may be unable to perform its control tasks as it is vulnerable to a remote code execution. There are no simulations and no failed queries in this system model. The runtime of this security analysis is 8.08 seconds on average, timed across 500 runs.

STEP 3: Performing Simulations.

A final step is to look for solutions and reason about other possible incidents through the use of simulations. There were two component and module vulnerabilities in the system, one concerning the setter PLCs and one concerning the SCADA PC operating system. No firmware update is available to fix the PLC issues, hence the assessor must find a new equipment vendor to replace his hardware. To choose between several possible stakeholders, the data flow analysis described in Section 4.3 can be used. The assessor can already simulate new types of PLCs to check if they introduce other vulnerabilities. For instance, when modelling the Saia PCD1.M2120 PLC, no more PLC vulnerabilities are reported, hence this is an option. For the OS, it suffices to apply the necessary patches. Updating the version number accordingly in the internal block diagram of the SCADA PC results in no more vulnerabilities.

Additional simulations can be run to investigate what happens in specific vulnerability scenarios. Components and modules can be assigned one of the five vulnerability categories to simulate what would happen if they contained a known vulnerability of that type. For instance the assessor might wonder what happens when the control software module inside the touch screens contains a vulnerability of type Denial of Service. To simulate this, it suffices to add a SysML property called "DoS" of type Vulnerability inside the touch screen block and connect it to the appropriate module, as shown in Figure 9. Now when the security analysis is ran again, the formatted output shows that a simulation was performed, as shown in Figure 10. The result of the denial of service is that the control module of the touch screen can no longer control its parameters. Further, this means the operator no longer has a means to modify parameters S_{1V} and S_{1UB}, which violates a query. The violated queries are shown at the bottom together with a set of parameters that causes them to fail. More information about the modelling and evaluation of simulations can be found at [19].

4.3 Performing a Data Flow Analysis

A second feature of the FAST-CPS framework is the data flow analysis of the system. This functionality applies to the entire system architecture. The analysis will return which assets can be stored or processed by the actors in the system, as well as any conflicts this may cause with the asset preferences of actors.

STEP 1: Choosing Actors.

Each cyber-physical system contains multiple stakeholders, for instance the equipment vendor, the customer, the company, etc. These stakeholder roles can be filled by different actors, i.e. the equipment vendor could be Siemens, Rockwell Automation, or another company. Each of these actors may have different data asset preferences and policies. Some may not want it possible to link a specific data asset to their identity by another stakeholder. Others may not want their data stored inside a cloud. The first task for the system owner is to choose the actors he wants to consider for each stakeholder role. Once those have been chosen, their asset preferences and policies must be collected. The FAST-CPS framework does not offer support for this step. These preferences and policies will be modeled as logic predicates and added to the framework as shown in STEP 2 below.

For the hatchery case study, the following stakeholders are considered: the *customer* who places orders, the *company* who runs the business side of the hatchery, the *process owner* who is in charge of the machines, the *equipment vendor* who provides equipment, and then two cloud stakeholders: one that takes care of the *scheduling* and one for the *predictive maintenance*.

For each stakeholder, multiple actors can be considered. The assessor is looking for a new equipment vendor due to the vulnerabilities encountered in the previous section. Three actors, *VendA*, *VendB* and *VendC* are under consideration. Further, the assessor can choose between *SAP HANA* and *Azure* for the scheduling cloud actor, and between *SAP HANA* and *AWS EC2* for the PdM cloud actor. Finally, the assessor has modelled two different customer actors representing different kinds of data asset preferences the customers tend to have, *CustA* and *CustB*.

Table 1 shows some actor *operation preferences* related to the data assets that are applicable to the hatchery case study. Operation preferences specify which stakeholders should be able to perform operations on data assets. Operations currently supported are Store (S) and Process (P). Customer A is worried about the privacy of his assets and only wants to reveal these assets when it is strictly necessary. When he places an order, only the company stakeholder should be able to store/process this information, the operators in the process network do not need to see this. Likewise, their personal information should not travel further than the company network. Customer B is less worried about his privacy and does not mind if the process operators have access to this data. Equipment vendor A does not want the process network to have access to the PdM sensor data, otherwise the operators could take matters in their own hands. Vend A also does not want the hatchery to have access to the code of their patches.

Table 1. Actor operation preferences.

Actor	Asset	Process	Comp	Vend	SCloud	PCloud
Cust A	*Order*		SP		P	
	CustomerName		SP			
Cust B	*Order*	SP	SP		P	
	CustomerName	SP	SP		P	
Vend A	*SensorData*			SP		P
	PatchCode			SP		
Vend B	*SensorData*	SP		SP		P
	PatchCode	SP		SP		
	OperatorLogs	SP		SP		
Vend C	*SensorData*	SP		SP		P
	PatchCode	SP		SP		
Hatchery	*SensorData*	SP		SP		P
	PatchCode	SP		SP		
	OperatorLogs	SP				

Equipment vendor B trusts the process network with both the sensor data and the patch code, in return they would like to have access to the operator logs to make the PdM process easier. Vendor C trusts the hatchery with sensor data and patch code and does not require additional logs. The hatchery actor wants access to both the sensor data and the patch code and has indicated this in the preferences. In addition, they do not want to release the operator logs to the equipment vendor for privacy reasons.

It is also possible to reason about *linkability* and *identifiability* preferences. For instance a customer could indicate that it is ok for a stakeholder to have access to both his Order and CustomerName, as long as the stakeholder is not able to *link* these to each other. For identifiability preferences, each actor can mark a set of assets as *identifiable*, indicating that this set of assets uniquely identifies them. They can then indicate that they do not want other stakeholders to be able to link assets to their identity. More information about how these preferences are modelled can be found at [18].

For the cloud actors, the assessor must also acquire their data policies and which directives they abide to. For instance, SAP HANA abides to the EU DPD whereas Azure complies with the Patriot act.

Finally, the assessor wants to consider two different flows of performing the predictive maintenance process. One where LoRa sensors are used and the sensor data is transported straight to the equipment vendor via a *TelCo* stakeholder, and one where the process owner is in charge of collecting and sending this data. All combinations of actors will be mapped on both flows to see which scenario creates the least conflicts.

STEP 2: Expanding the System Model.

When the necessary information is collected, the assessor must update the IDP input file with it. The input.idp file is a result of the security analysis

performed in the previous section. It already contains information regarding the components, channels and processes that are present in the system. Now the assessor must manually expand the file to include stakeholders, actors and assets as follows:

$$\texttt{type}\ \text{Stakeholder} = \{\mathit{Customer}; \mathit{EquipmentVendor}; \mathit{Company}; \ldots\}$$
$$\texttt{type}\ \text{Actor} = \{\mathit{CustA}; \mathit{CustB}; \mathit{VendA}; \ldots\}$$
$$\texttt{type}\ \text{Asset} = \{\mathit{Order}; \mathit{CustomerName}; \mathit{MachineID}; \mathit{MachineProblem}; \ldots\}$$

Next, the operator preferences are added as tuples to a predicate:

$$\mathsf{OperationPref}(\text{Actor}, \text{Asset}, \text{Stakeholder}, \text{AssetOperation}) = \{ (\mathit{CustA}, \mathit{Order}, \mathit{Company}, S); (\mathit{CustA}, \mathit{Order}, \mathit{SCloud}, P); (\mathit{CustA}, \mathit{CustomerName}, \mathit{Company}, S); \ldots\}$$

The data policies and directives are added in similar fashion:

$$\mathsf{Directive}(\text{Actor}, \text{Directive}) = \{ (\mathit{SAPHANA}, \mathit{EUDPD}); (\mathit{Azure}, \mathit{Patriot}); (\mathit{AWSEC2}, \mathit{Patriot})\}$$

Finally, the asset flow must be modelled, showing which assets flow between which components in the system, and where they are processed. This is done by indicating the sender component, receiver component, and the particular assets that are sent:

$$\mathsf{AssetFlow}(\text{Component}, \text{Asset}, \text{Component}) = \{ (\mathit{CustPC}, \mathit{Order}, \mathit{CustRouter}); (\mathit{CustRouter}, \mathit{Order}, \mathit{CompanyRouter}); (\mathit{CustPC}, \mathit{CustomerName}, \mathit{CustRouter}); \ldots\}$$

To show the feasibility of the approach, two different flows are modelled, resulting in two different IDP input files. In the first flow, a LoRa sensor is used for predictive maintenance, which means the *SensorData* asset is sent directly to the equipment vendor stakeholder over a TelCo network. In the second flow, this sensor data first passes through the SCADA PC in the process network, and is then forwarded by the operators to the equipment vendor.

Adding tuples to the IDP file like this is tiresome and error-prone. A GUI will be provided to enhance the user experience when performing the data flow analysis. However, it will still be the assessor's responsibility to provide a complete model, the logic framework cannot check for completeness.

STEP 3: Performing the Data Flow Analysis.

Once the IDP input model has been extended, IDP can be run again to perform the data flow analysis of the CPS. The output is two-fold: IDP will fill an *AssetProfile* predicate which will show the operations that stakeholders can perform on assets. This way the assessor can get a full overview of what happens to the data assets in the system. The analysis will then return all conflicts that exist between this profile and the preferences provided by the actors. The engine will automatically try all possible combinations of actors and return only the models that provide the least amount of conflicts. Currently no suggestions on how to fix these conflicts are generated by the methodology. In this article the analysis is performed to show the feasibility of the methodology, for details on how the conflicts are found and a look of what the logic engine looks like, the reader is referred to [18].

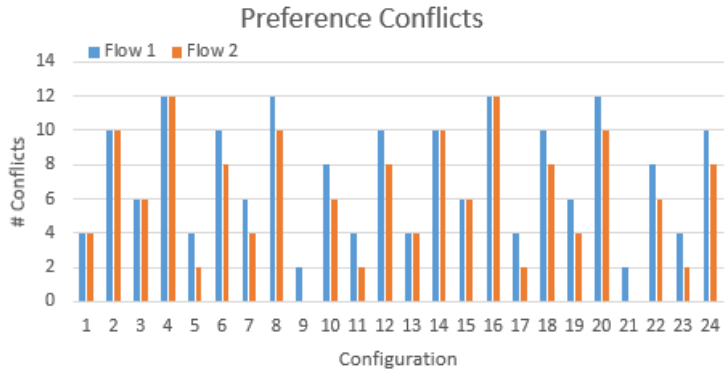

Fig. 3. The number of conflicts returned for each configuration.

In our case study, there are 24 possible configurations of actors for each flow. Figure 3 shows the number of conflicts returned for each configuration for both flows. It can immediately be observed that the second flow should be chosen. For the first flow which uses the LoRa sensor, two models are returned that each have 2 conflicts with the preferences. In both cases, these conflicts are caused by the fact that the Hatchery actor does not gain access to the sensor data. The flow without LoRa contains two models with 0 conflicts. The actors selected in these two instances are (CustA, VendC, SAPHANA, SAPHANA) and (CustB, VendC, SAPHANA, SAPHANA). Hence the assessor can conclude that he should use VendC as the new equipment vendor and SAPHANA for the cloud operations. The runtime of the data flow analysis is 9.06 seconds on average, timed across 500 runs.

STEP 4: Generating Additional Feedback.

When there are no cases with 0 conflicts, the assessor still has to make decisions about which actors to select. FAST-CPS contains an additional Java

program to further analyse the IDP results. When given an IDP file and a specific actor, the Java program will try all possible combinations of actors, return the number of conflicts in each configuration, and then also display the number of conflicts caused by the given actor's preferences. If one actor's preferences cause more conflicts, the assessor can opt for the other.

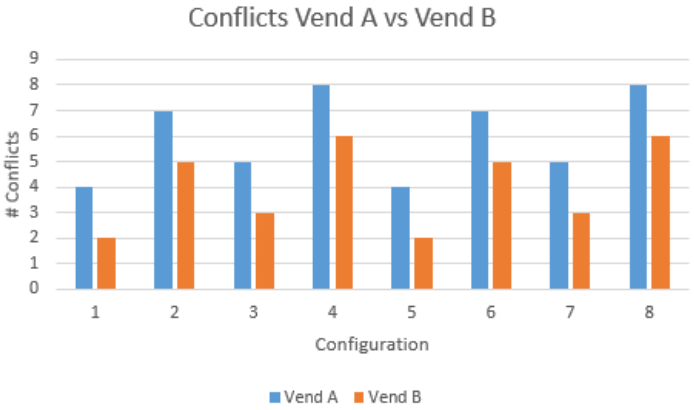

Fig. 4. The number of conflicts caused by equipment vendor A vs B.

For example, if Vend C were not available and the assessor had to choose between vendors A or B, he could plot their conflicts in the same configurations as shown in Figure 4. As shown, vendor B would be the better option in this case, as he causes less conflicts in all scenarios.

Currently all conflicts are considered equal, but in reality some conflicts will be more severe than others. In future work the logic engine will be extended to allow actors to assign a weight to their preferences. Based on these preference weights, the conflicts would also be weighted and the severity of conflicts will also be taken into account.

4.4 Towards a Qualitative Risk Assessment

A final feature of the FAST-CPS framework is the automatic construction of attack trees which can then be evaluated to reason about an attacker's optimal attack strategy. Based on the IDP output file generated in part 1, an attack tree is automatically constructed for a chosen attacker goal. The assessor can then model an attacker in more detail than in part 1 and evaluate the tree with respect to this attacker. The result will be an optimal path for this attacker, where a path is deemed optimal if it is the least difficult road to reaching the goal. This information can then be used to perform a qualitative risk assessment after the assessor assigns impact values to the attacker goals.

STEP 1: Generating an Attack Tree.

The FAST-CPS framework contains an additional Java program which is used for the attack tree creation and evaluation. This program can be found at [15]. When launched, the assessor is prompted to provide a FAST-CPS output file to build the tree from. This file is the result of the security evaluation in section 4.2. After choosing this file, the assessor must also provide the attacker goal for which he wants to build an attack tree. Three main types of attacker goals were identified based on the NIST guide to industrial control system security [32]:

- Modifying a process parameter p
- Launching a denial of service on a hardware component or software module s
- Obtaining data asset d

Once the attacker has chosen a goal and provided the relevant system parameter p, s or d, the tree is automatically constructed with the chosen goal as the root node. The tree generation algorithm uses templates that represent attacks on the system or assess the impact of vulnerable components on possible threats. The templates can be found at [16], the algorithm is shown below:

GenerateTree(AttackerGoal,SystemModel,Templates):

```
tree.root = AttackerGoal
goals = { AttackerGoal }
while goals ≠ ∅ do
    goal = goals.pop()
    foreach Template t ∈ Templates do
        if t.goal = goal then
            Tree s = t.execute(goal, SystemModel)
            goals.push(s.leaves)
            tree.replace(goal, s)
```

In the hatchery case study, the assessor wants to reason about two different kinds of attacks, performed by different attackers. First of all, the assessor wants to know how easy it would be for malicious employees in the company or equipment vendor network to perform a denial of service attack on their process. Hence the first tree that will be built has as root goal $DoS(Setter1PLC)$. When constructing this tree, it contains 13 nodes. The tree is shown in Figure 11.

The second goal concerns the modification of process parameters. The assessor wants to make sure that technicians and any outside attackers are unable to change the incubator valves. The tree constructed for $ModifyParameter(S_{1V})$ contains 135 nodes. This tree is built in an averaged runtime of 0.26 seconds

STEP 2: Modelling the Attacker.

When the tree is created, the assessor must model the attackers which he will evaluate the tree for. An Attacker $\mathcal{A}$ *= (C, M, A)* is defined by the set of credentials "C" the attacker owns, a mapping of *attacker capabilities* to integer values "M", and a set of components in the system "A" which the attacker has physical access to. The following attacker capabilities are considered:

- *Stealing Credentials*: Stealing credentials from other users in the system by performing a phishing attack, social engineering, etc.
- *Identity Spoofing*: Pretending to be another entity whilst sending commands over a channel.
- *Exploiting Vulnerabilities*: Exploiting known vulnerabilities in components or modules.
- *Discovering Vulnerabilities*: Discovering zero-day vulnerabilities in components or modules.

If an attacker is chosen that was modelled inside the SysML model, sets C and A will be constructed automatically by looking at the FAST-CPS file. If the assessor wants to consider a new outside attacker with different properties, these credentials and components must be added manually using the "Model Attacker" button in the program. Here the attacker capabilities are also modelled by giving a number between 1 (no prior expertise) and 4 (expert in the subject) to each capability.

Table 2. Attacker capabilities

Capability	Technician	Outsider	Company	Vend
Stealing Credentials	2	3	1	3
Identity Spoofing	2	4	1	2
Exploiting Vulnerabilities	4	4	2	3
Discovering Vulnerabilities	1	3	1	1

Four types of attackers are relevant for the industrial hatchery case study. One of them is an outside attacker, the others are disgruntled employees: a technician in the control network, an employee in the company network, and an employee of the equipment vendor company.

The technician has access to all the components inside the process network, but he does not possess any credentials. The outside attacker has no access and no credentials. The employees of the company and the equipment vendor have access to the components inside their respective networks, and do not own any credentials from the process network. In addition, the employees from the equipment vendor can connect remotely to the SCADA PC in the process network through the EWON router placed there. The capabilities of the attackers are shown in Table 2.

STEP 3: Evaluating the Attack Tree.

Once the attacker is modelled and an attack tree is generated, the tree can be used to analyse security properties of the system. For now only the difficulty of attacks is evaluated, other heuristics will be added in the future. Difficulties range between 1 (Trivial) and 4 (Unlikely). The analysis proceeds in three steps: First the difficulty of each leaf is determined, then the difficulty of parent nodes is synthesized from the difficulty of their children. Once all nodes are annotated,

the optimal attacker strategy is calculated. An optimal attacker strategy is a subtree of the attack tree which represents the optimal way for a modelled attacker to reach the attacker goal. What is optimal depends on the chosen heuristic in the tree evaluation stage, here it will be the path that results in the lowest difficulty of achieving the attacker goal. Once the optimal attacker strategy is returned, the assessor can identify the most vulnerable parts of the system by looking at the leaves of the path.

Assigning Values to Leaves. Tree leaves will either be associated with an attacker's capabilities, an attacker's access to components, or an attacker's credentials. Once a particular type of attacker has been chosen, the difficulties are decided as follows:

- Credential leaf: If the attacker possesses the credential, the difficulty is 1, if he does not, the difficulty is 4.
- Access leaf: If the attacker has physical access to the component, the difficulty is 1, if he does not, the difficulty is 4.
- Capability leaf: Here the algorithm assigns a difficulty based on the capability value the attacker was given for the relevant skill. Capability values range between 1 and 4: 1 means no prior expertise, 2 means little expertise, 3 means moderate expertise and 4 means the attacker is an expert in the subject. A capability value of 1 is mapped with a difficulty of 4, 2 with a difficulty of 3, etc.

Propagating Values up the Tree. Now each leaf has a difficulty d, as well as a value s which represents the least amount of steps that must be taken to reach it, which is 1 for leaves. These values are now propagated up the tree using the algorithms below. Both algorithms are called on the root of the tree.

CalculateDifficulty(n):

If n is a Leaf **then** return $n.d$ **else**
If $n.o = \wedge$ **then** return Max $\{CalculateDifficulty(c) | c \in n.C\}$.
If $n.o = \vee$ **then** return Min $\{CalculateDifficulty(c) | c \in n.C\}$.

CalculateSteps(n):

If n is a Leaf **then** return 1 **else**
If $n.o = \wedge$ **then** return Sum $\{CalculateSteps(c) | c \in n.C\}$.
If $n.o = \vee$ **then** return Min $\{CalculateSteps(c) | c \in n.C \wedge c.d$ is lowest among all children$\}$.

Calculating the Optimal Attacker Strategy. Once all the nodes in the tree have values for the pair (d, s), the optimal attacker strategy can be calculated. The heuristic for this can vary between implementations, the algorithm outlined below prioritizes strategies with the lowest difficulty and uses s as a tie-breaker: $(d, s) < (d', s') \Leftrightarrow (d < d') \vee (d = d' \wedge s < s')$. All nodes that are part of the

optimal strategy will have their flag f set to true. A new tree S can be generated using these nodes and the edges that connect them. The algorithm is called on the root of the tree:

OptimalAttackerStrategy(n):

Set flag $n.f$ to true.
If n has no children **then** end **else**
If $n.o = \wedge$ **then** call OptimalAttackerStrategy(c) for each node $c \in n.C$.
If $n.o = \vee$ **then** call OptimalAttackerStrategy(c) on node $c \in n.C$ such that $\forall\ c' \in n.C$: $(c.d, c.s) < (c'.d, c'.s)$

Table 3. Attack difficulties

Attack Goal	Technician	Outsider	Company	Vend
$DoS(Setter1PLC)$	N/A	N/A	4	2
$ModifyParameter(S_{1V})$	3	4	N/A	N/A

The results for the case study are shown in Table 3. For employees of the company network, performing a DoS attack receives the highest possible difficulty: 4. The employees have no way of accessing the process network, which makes the attack impossible. The employees from the equipment vendor can access the process network remotely through the EWON router. This allows a malicious employee to try and exploit the DoS vulnerability of the PLC. Hence the difficulty for this attacker depends on his capability for exploiting vulnerabilities. A capability of 3 corresponds to an attack difficulty of 2.

For the other attack, the outside attacker has difficulty 4 due to the lack of access to the process network. The technicians have access to the network where the operator passwords are stored, hence the difficulty of modifying a parameter depends on their capability of stealing the necessary credentials. In the case of the technician this means the attack difficulty is 3 as their capability is 2.

Based on the optimal attack strategy, i.e. the path in the tree that is coloured red, the assessor can now identify the weak points in his system and attempt to make it more secure.

5 Evaluation

In this section the FAST-CPS framework will be evaluated. The main purpose of the framework is to provide useful feedback to system operators about the security of their system, without requiring too much effort or security expertise on the side of the operator. These points will be evaluated, as well as the scalability and extensibility of the framework, and the soundness and completeness of the feedback.

For using FAST-CPS, the main assessor effort lies in modelling the system in the SysML modelling language. Finding the correct balance between required input and usefulness of the returned feedback was one of the main challenges of implementing the FAST-CPS framework. Currently the methodology makes abstraction of some system elements, e.g. any communication protocols are not modelled. The aim was to reduce the modelling effort as much as possible whilst still being able to extract the desired feedback. As a result, even for more complex systems it should not take more than a full day to model them using the SysML approach. The use case described in the article was modelled in a matter of hours. Possible changes to the system architecture can easily be incorporated in the SysML model and do not require a full remodelling of the system. In related tools, the assessor must define the workflow of the attacker as well, which is no trivial task and security expertise is required to know how an attacker would most likely go about attacking the system. For modelling a system in FAST-CPS, no security expertise is required.

FAST-CPS offers multiple kinds of feedback that are not provided by other tools. Using CPS vulnerability databases, known vulnerabilities in the hardware and software are found and their effects on the system processes are calculated. This process is fully automated and other than a system model the user must not provide any input. This feedback is useful for assessors as the alternative is to repeatedly check these vulnerability databases themselves. Also, the databases do not show the impact of the vulnerabilities on the system processes, while the framework does. Next, FAST-CPS helps assessors decide on which actors to use for stakeholders roles in their system by showing them which configurations of actors cause the least conflicts with data asset preferences. An asset profile shows exactly what happens to data assets in the system, which is no longer trivial to know in today's complex system architectures. Finally, attack trees are created automatically and show how attackers will be most likely to attack the system. Hence the attacker workflow which is input for related tools is provided as output by FAST-CPS. Both for the creation of the asset profile and for generating the attack trees, the vulnerability information extracted during the first part of the framework is used, which means the feedback is not trivial to find manually by an assessor.

The framework does not suggest solutions on how to fix the issues in the system. However by inspecting the leaves of an optimal attack path the assessor can get a good idea of what the weak points in the system are, and through the use of simulations he can change the model and try several solutions until a new evaluation eliminates the previous shortcomings. Since the source of the feedback are known vulnerability databases, the framework also does not help to protect against zero-day vulnerabilities. Here it is strongly suggested to use the framework in combination with other security measures such as an intrusion detection/prevention system.

The inclusion of vulnerability databases also results in a drawback. Every time new vulnerabilities are added to the databases, these must also be added to the framework. Currently this is done manually by the FAST-CPS adminis-

trators. If the framework is to be released for commercial use, a better option must be provided. One solution could be to run the framework as a software application on-site that can be connected to remotely. The FAST-CPS administrators could then connect to the software and push any required updates as software patches. However, there are drawbacks to this approach, one of which is that the assessor must place trust in the FAST-CPS administrators. In addition, all updates would have to be pushed to every single FAST-CPS instance. A better solution would be to place the software in the cloud. Now the assessors of various CPS must connect to the same software instance in order to scan their systems. This means the administrators must only keep one version of the software up to date, rather than many. An additional benefit of using the cloud is that there cannot be any interference between the FAST-CPS framework and the industrial network as they remain separated.

In terms of scalability, both SysML and IDP are capable of modelling large systems. In SysML, the assessor can continue to create new diagrams to expand a system, whereas in IDP the structure can continue to be extended with new tuples. In terms of evaluation, extracting vulnerabilities and computing their impact on the system processes is a model expansion problem which means the runtime will not increase significantly as the system gets bigger. The data flow analysis is a search problem and here the runtime will start to slow down as a large number of actors is modelled for each stakeholder in the system. It should be noted that this evaluation has no real-time requirements and it is acceptable for the data flow analysis to run multiple minutes in a worst-case scenario. For the case study in this chapter, which presents a full-scale, real cyber-physical system, the vulnerability extraction took on average 3.64 seconds while the data flow analysis took on average 9.06 seconds, timed across 100 runs.

As the framework is an extension of the Papyrus Eclipse environment, it is possible to extend it further and add new functionality to it. New Eclipse plug-ins can be written that add additional menu options to the framework and provide new security analysis functions.

Related to soundness and completeness of the feedback, two important questions can be asked:

- Is the feedback *complete*, i.e. does the framework find all vulnerabilities in the system?
- Is the feedback *sound*, i.e. if the framework finds a vulnerability in the system model, is it actually present in the system?

The answers to these questions depend on the scope of the methodology, the accuracy of the model and whether the logic engine has been updated to contain the latest entries of each vulnerability database:

The methodology presented in this thesis is focussed on identifying known vulnerabilities and their impact on the system processes. Provided the logic engine is up to date in terms of vulnerability databases, the framework will extract any known vulnerabilities from the system model and accurately assess the impact of these vulnerabilities on the system processes. Zero-day vulnerabilities

and their impact on the processes are outside the scope of the methodology and will not be identified. To this end, it is advised to combine the FAST-CPS framework with an intrusion prevention or detection system that is capable of picking up these additional threats.

If a system vulnerability is inside the scope of the methodology, the vulnerability databases have been updated in the logic engine, and the system has been modelled correctly by the assessor, the system vulnerability will be found by the FAST-CPS framework, so in this sense the feedback is only complete inside the defined scope of the methodology. If any of the previous conditions are not met, the vulnerability may not be picked up. This implies that a system is not necessarily secure when the feedback returns zero vulnerabilities. Recall that this work assumes that any unused ports are hardened, no unnecessary software is present on workstations or HMIs, firewalls are correctly configured, etc. The framework cannot ascertain whether this is in fact the case, which means there could still be vulnerabilities in the system even when the methodology returns none. In order to protect the system against these vulnerabilities, it is recommended to combine FAST-CPS with other security measures. A full system audit should still be performed on a timely basis in order to update the system inventory and to evaluate the security policies and procedures, which FAST-CPS does not provide feedback about. An intrusion detection or prevention system can be set up alongside FAST-CPS to monitor the network traffic inside the system. If any firewalls are wrongly configured or malware has found its way inside the network, an IDS/IPS could detect and prevent this and keep the system secure.

Regarding soundness, when a vulnerability is found in the system model it will also be present in the actual system. There are two exceptions to this statement:

- When the vulnerability was the result of a simulation. In this case the vulnerability only occurs on the system model and not in the actual system. The output file will always clearly state which simulations the assessor has included in the model.
- When the assessor has incorrectly modelled his system. In this case the feedback provided by the framework may not apply to the actual system. The assessor has the responsibility of correctly modelling his system, guidelines on how to do this in SysML are provided with the framework.

When the system has been modelled correctly and a returned vulnerability is not the result of a simulation, the vulnerability will also be present in the system, hence the feedback provided by our methodology is sound.

6 Conclusions

This article illustrates the use of the FAST-CPS framework for analysing the security of cyber-physical systems. The framework is introduced and compared to other security assessment tools. Then the framework is used to analyse an industrial hatchery. The necessary steps that must be taken by the assessor are

explained in detail and the different kinds of feedback returned by the framework are presented.

Appendix A FAST-CPS Screenshots

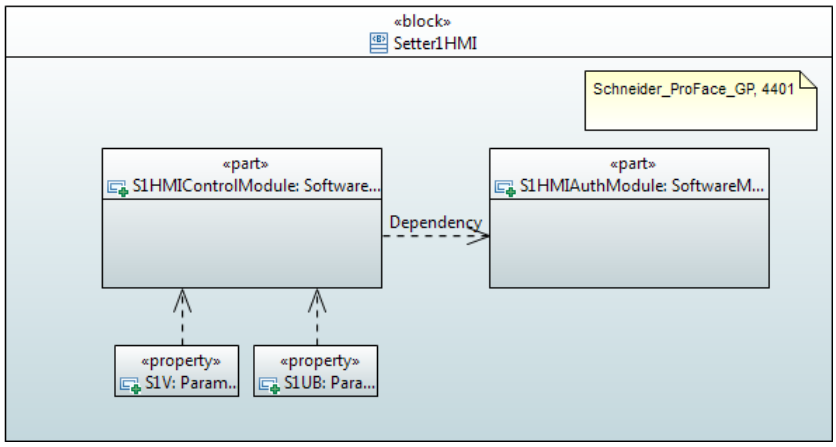

Fig. 5. The internal block diagram representing one of the setter touch screens.

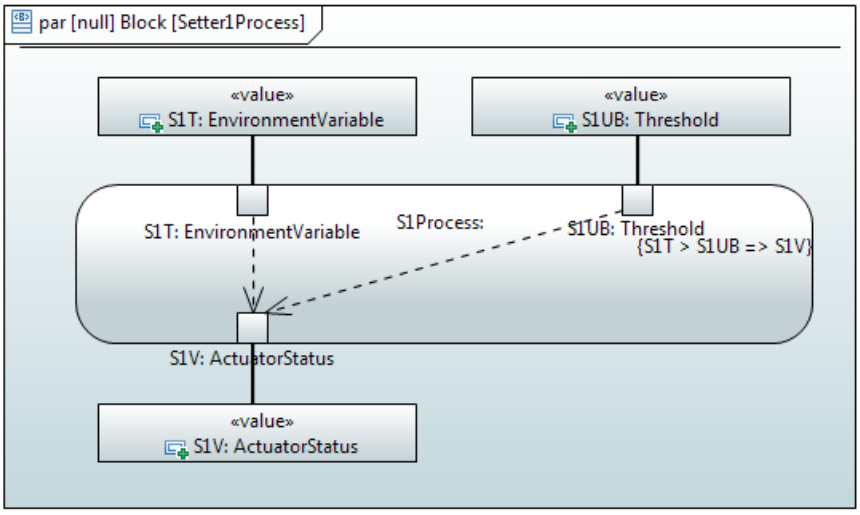

Fig. 6. The parametric diagram representing the temperature process in the setter.

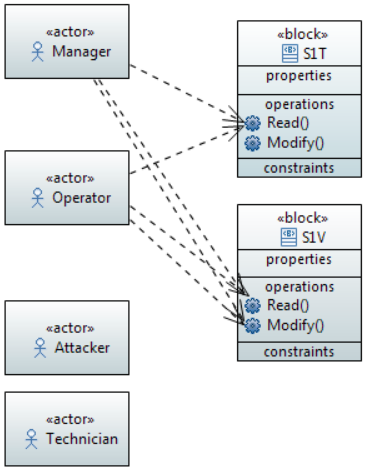

Fig. 7. The block diagram representing the policy specification of the system.

*model.di input.idp output.idp output.txt

```
//-------------------------------------------------------------------------------
// Component Vulnerabilities
//-------------------------------------------------------------------------------
"SCADAPCOSModule" has a RemoteCodeExecution vulnerability
"Setter1PLC" has a DataLeakage vulnerability
"Setter1PLC" has a DoS  vulnerability

Simulations:


//-------------------------------------------------------------------------------
// Compromised System Tasks
//-------------------------------------------------------------------------------

Compromised Storage tasks:

"Setter1PLC" leaks parameter "S1T"
"Setter1PLC" leaks parameter "S1UB"
"Setter1PLC" leaks parameter "S1V"

Compromised Control tasks:

"SCADAPCControlModule" cannot change the status of "H1UB"
"SCADAPCControlModule" cannot change the status of "H1V"
"SCADAPCControlModule" cannot change the status of "S1UB"
"SCADAPCControlModule" cannot change the status of "S1V"

Compromised Communication tasks:

"Setter1TempSensor" does not deliver "S1T" to component "Setter1PLC"

//-------------------------------------------------------------------------------
// Failed Queries
//-------------------------------------------------------------------------------
```

Fig. 8. The result of the first security analysis.

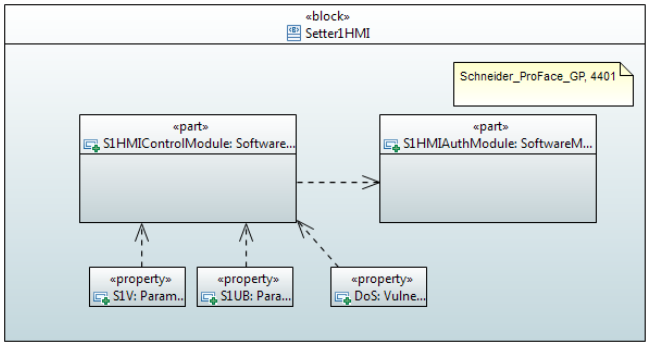

Fig. 9. Modelling a denial of service simulation of a software module.

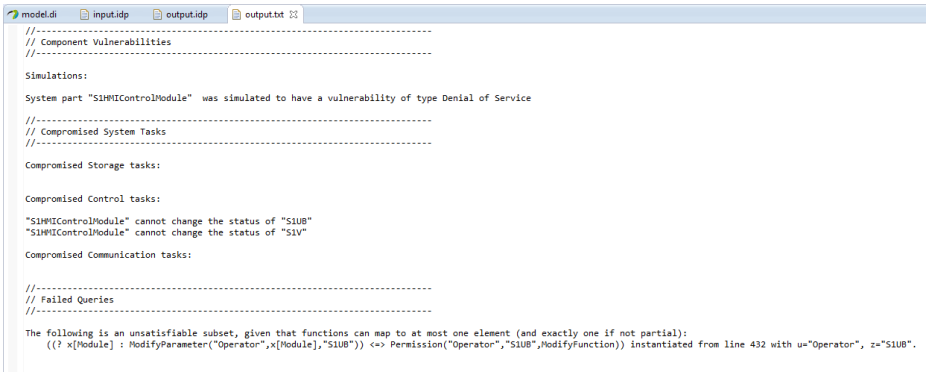

Fig. 10. The result of the second security analysis using simulations.

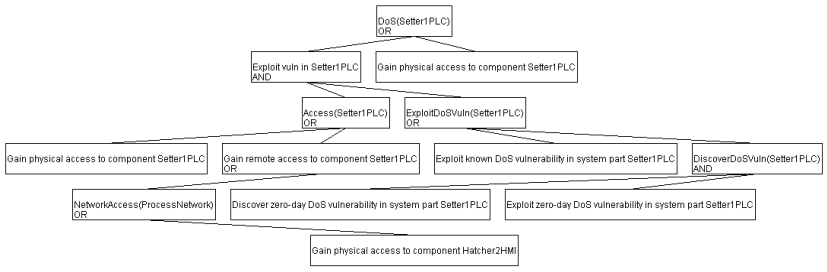

Fig. 11. The attack tree constructed for DoS(Setter1PLC).

Acknowledgements

Research funded by a PhD grant of the Agency for Innovation by Science and Technology in Flanders (IWT).

References

1. Marshall Abrams and Joe Weiss. Malicious control system cyber security attack case study–maroochy water services, australia, 2008.
2. Shweta Agrawal and Dan Boneh. Homomorphic macs: Mac-based integrity for network coding. In *International Conference on Applied Cryptography and Network Security*, pages 292–305. Springer, 2009.
3. Michael Assante. Confirmation of a coordinated attack on the ukrainian power grid, 2016.
4. Jonathan P Chapman, Simon Ofner, and Piotr Paulksztelo. Key factors in industrial control system security. In *Local Computer Networks (LCN), 2016 IEEE 41st Conference on*, pages 551–554. IEEE, 2016.
5. Qian Chen, Robert K Abercrombie, and Frederick T Sheldon. Risk assessment for industrial control systems quantifying availability using mean failure cost (mfc). *Journal of Artificial Intelligence and Soft Computing Research*, 5(3):205–220, 2015.
6. ENISA. Protecting industrial control systems: Recommendations for europe and member states, 2011.
7. Nicolas Falliere, Liam Murchu, and Eric Chien. W32.Stuxnet Dossier, 2011.
8. Michael D Ford, Ken Keefe, Elizabeth LeMay, William H Sanders, and Carol Muehrcke. Implementing the advise security modeling formalism in möbius. In *Dependable Systems and Networks (DSN), 2013 43rd Annual IEEE/IFIP International Conference on*, pages 1–8. IEEE, 2013.
9. Virgil Dorin Gligor and Pompiliu Donescu. Block encryption method and schemes for data confidentiality and integrity protection, December 6 2005. US Patent 6,973,187.
10. Sari Stern Greene. *Security policies and procedures.* New Jersey: Pearson Education, 2006.
11. http://ics-cert.us-cert.gov/Assessments Homeland Security. Cset: Cyber security evaluation tool., 2014.
12. ICS-CERT. Rockwell automation controllogix plc vulnerabilities (update a), 2014.
13. Ralph Langner. To kill a centrifuge: A technical analysis of what stuxnet's creators tried to achieve, 2013.
14. Edward A Lee. Cyber physical systems: Design challenges. In *Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on*, pages 363–369. IEEE, 2008.
15. Laurens Lemaire. Sysml modelling instructions, `https://github.com/LaurensZJL/`, 2017.
16. Laurens Lemaire. Tree templates and tool, `https://github.com/AttackTrees/`, 2017.
17. Laurens Lemaire, Jan Vossaert, Bart De Decker, and Vincent Naessens. An assessment of security analysis tools for cyber-physical systems. In *Proceedings of the 4th International Workshop on Risk Assessment and Risk-Driven Quality Assurance*, 2017.

18. Laurens Lemaire, Jan Vossaert, Bart De Decker, and Vincent Naessens. Extending fast-cps for the analysis of data flows in cyber-physical systems. In *Proceedings of the 7th International Conference on Mathematical Methods, Models and Architectures for Computer Network Security*, 2017.
19. Laurens Lemaire, Jan Vossaert, Joachim Jansen, and Vincent Naessens. A logic-based framework for the security analysis of industrial control systems. *Automatic Control and Computer Sciences*, 51(2):114–123, 2017.
20. Elizabeth LeMay, Michael D Ford, Ken Keefe, William H Sanders, and Carol Muehrcke. Model-based security metrics using adversary view security evaluation (advise). In *Quantitative Evaluation of Systems (QEST), 2011 Eighth International Conference on*, pages 191–200. IEEE, 2011.
21. Neil MacDonald. Devops needs to become devopssec, `https://blogs.gartner.com/neil_macdonald/2012/01/17/devops-needs-to-become-devopssec/`.
22. Aleksandr Matrosov, Senior V. Researcher, Eugene Rodionov, Rootkit Analyst, and David Harley. Stuxnet Under the Microscope, 2011.
23. Stephen McLaughlin, Charalambos Konstantinou, Xueyang Wang, Lucas Davi, Ahmad-Reza Sadeghi, Michail Maniatakos, and Ramesh Karri. The cybersecurity landscape in industrial control systems. *Proceedings of the IEEE*, 104(5):1039–1057, 2016.
24. Microsoft. Microsoft security bulletin ms17-010, 2017.
25. Savita Mohurle and Manisha Patil. A brief study of wannacry threat: Ransomware attack 2017. *International Journal*, 8(5), 2017.
26. Clifford Neuman. Challenges in security for cyber-physical systems. In *DHS Workshop on Future Directions in Cyber-Physical Systems Security*, pages 22–24. Citeseer, 2009.
27. Robert Oates, Fran Thom, and Graham Herries. Security-aware, model-based systems engineering with sysml. In *Proceedings of the 1st International Symposium for ICS & SCADA Cyber Security Research*, page 78, 2013.
28. Ganesh J Pai and J Bechta Dugan. Automatic synthesis of dynamic fault trees from uml system models. In *Software Reliability Engineering, 2002. ISSRE 2003. Proceedings. 13th International Symposium on*, pages 243–254. IEEE, 2002.
29. Ahmad-Reza Sadeghi, Christian Wachsmann, and Michael Waidner. Security and privacy challenges in industrial internet of things. In *Proceedings of the 52nd Annual Design Automation Conference*, page 54. ACM, 2015.
30. Teodor Sommestad, Mathias Ekstedt, and Hannes Holm. The cyber security modeling language: A tool for assessing the vulnerability of enterprise system architectures. *Systems Journal, IEEE*, 7(3):363–373, 2013.
31. Teodor Sommestad, Mathias Ekstedt, and Lars Nordström. A case study applying the cyber security modeling language, 2010.
32. Keith Stouffer, Suzanne Lightman, Victoria Pillitteri, Marshall Abrams, and Adam Hahn. Guide to industrial control systems (ics) security, 2015.
33. An Hoa Vu, Nils Ole Tippenhauer, Binbin Chen, David M Nicol, and Zbigniew Kalbarczyk. Cybersage: a tool for automatic security assessment of cyber-physical systems. In *International Conference on Quantitative Evaluation of Systems*, pages 384–387. Springer, 2014.
34. Martin Walker, Mark-Oliver Reiser, Sara Tucci-Piergiovanni, Yiannis Papadopoulos, Henrik Lönn, Chokri Mraidha, David Parker, DeJiu Chen, and David Servat. Automatic optimisation of system architectures using east-adl. *Journal of Systems and Software*, 86(10):2467–2487, 2013.

35. Eric Ke Wang, Yunming Ye, Xiaofei Xu, SM Yiu, LCK Hui, and KP Chow. Security issues and challenges for cyber physical system. In *Proceedings of the 2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing*, pages 733–738. IEEE Computer Society, 2010.
36. Kim Zetter. Inside the cunning, unprecedented hack of ukraine's power grid, 2016.